K2: A Stream Cipher Algorithm Using Dynamic Feedback Control

Shinsaku Kiyomoto[1], Toshiaki Tanaka[1], and Kouichi Sakurai[2]

[1] KDDI R & D Laboratories Inc.
2-1-15 Ohara, Kamihukuoka-shi, Saitama
356-8502, Japan
{kiyomoto, tl-tanaka}@kddilabs.jp
[2] Dept. of CSCE., Kyushu University
744 Motooka Nishi-ku, Fukuoka
812-0053, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. A variety of different clock-controlled stream ciphers and attacks on them have been described in a number of papers. However, few word-oriented algorithms with an irregular clocking mechanism have been proposed. This paper proposes a new design of irregular clocking for word-oriented stream ciphers that is dynamic feedback control and show analysis results of its security and performance. The stream cipher K2 v2.0 is a secure and high-performance stream cipher using the dynamic feedback control, which is applicable for several applications. We believe that the dynamic feedback control mechanism is potentially effective against several different types of attacks, not only existing attacks but also novel attacks.

keywords: Stream Cipher, Dynamic Feedback Control, Irregular Clocking, Pseudorandom Generator.

1 INTRODUCTION

Stream ciphers are used extensively to provide a reliable, efficient method for secure communications. A basic stream cipher uses several independent linear feedback shift registers (LFSRs) together with nonlinear functions in order to produce a keystream. The keystream is then XORed with plaintext to produce a ciphertext. Some stream ciphers use a general nonlinear function to clock one or more LFSR(s) irregularly. Various clock-controlled stream ciphers and attacks on them have been proposed. Clock-controlled stream ciphers are classified into two main types of stream ciphers: the A5 family and the LILI family. A5 is a well-known stream cipher designed to ensure the confidentiality of mobile communications. LILI-like stream ciphers, such as LILI-128 [Simpson et al., 2000],

have two different LFSRs for providing bits for clocking and keystream bits. One LFSR clocks regularly, providing input for a clock controller, and another LFSR clocks irregularly, providing keystream.

Recently, word-oriented stream ciphers have been developed in order to improve the performance of software implementations. In the NESSIE project, many word-oriented stream ciphers were proposed, such as SNOW [Ekdahl and Johansson, 2000] and SOBER [Rose and Hawkes, 1999], and demonstrated good performance in software. However, few word-oriented algorithms with an irregular clocking mechanism have been proposed because of the inefficiency of the clock control mechanism for software implementation. LFSR-based stream ciphers have been broken using a linear recurrence of the LFSR. Thus, an irregular clocking mechanism for word-oriented LFSRs will achieve an improvement of their security.

This paper proposes a new word-oriented stream cipher using dynamic feedback control as irregular clocking. The proposed stream cipher has a dynamic feedback control mechanism for the byte-level feedback function of FSRs and realizes fast encryption/decryption for software implementation. We present a stream cipher algorithm and show the results of an analysis of security and performance, and show that the cipher is secure and it offers high-performance encryption and decryption.

2 DYNAMIC FEEDBACK CONTROL

The clock control mechanism of a stream cipher generally either controls LFSR clocking or shrinks or thins output. A clock control that shrinks or thins output reduces the performance of the stream cipher because some output bits are discarded. If one applies shrinking to a word-oriented stream cipher, the performance is markedly reduced. The bit-oriented clock control mechanism for updating an LFSR is also inefficient when the mechanism controls the LFSR for each register. On the other hand, a dynamic feedback control for an LFSR is an effective method for improving the security of stream ciphers. The stream cipher MICKEY[Babbage and Dodd, 2005] has a dynamic feedback control mechanism for a bit-wise LFSR. POMARANCH[Jansen et al., 2005] uses a cascade jump controlled sequence generator to modify the feedback function.

We propose a stream cipher design (called KARAKORUM type) that operates on words and has an efficient dynamic feedback control as irregular clocking. The basic idea of the design is to modify the mixing operation during the state update. Feedback polynomials for word-oriented

LFSR are described with coefficients; multiplying an input word by a coefficient means mixing the words. A typical example is a LFSR of SNOW 2.0[Ekdahl and Johansson, 2003]. Generally, the coefficients are selected such that the feedback polynomial is a primitive polynomial. We apply irregular clocking for this mixing operation, and the modification causes only a minimal decrease in the encryption/decryption speed. In other words, at least one FSR is irregularly clocked to dynamically modify the feedback function to the dynamic feedback controller that receives the outputs of the other FSR(s). For example, the feedback function is defined as $s_{t+a} = \alpha_0^{\{0,1\}} s_{t+b} \oplus \alpha_1^{\{0,1\}} s_{t+c} \oplus \alpha_2^{\{0,1\}} s_{t+d}$, where $\{0, 1\}$s are selected by the dynamic feedback controller. The FSR controlled by the dynamic feedback controller is named dynamic feedback shift register (DFSR).

The dynamic feedback control mechanism improves the security of a stream cipher because it changes the deterministic linear recurrence of some registers into a probabilistic recurrence. This property effectively protects against several attacks. An attacker has to obtain the linear recurrence of the keystream derived from the linear recurrence of some registers. By an irregular modification, the linear recurrence exists with a low probability. An attacker has to guess some inputs to the non-linear function for an attack; however, an irregular modification makes it impossible: the attacker has to guess the inputs to the dynamic feedback controller first. Thus, irregular modification of the feedback function improves the security of the stream cipher.

We think that a dynamic feedback control mechanism is potentially effective against several attacks, not only existing attacks but also a novel attack.

3 STREAM CIPHER K2 V2.0

In this section, we describe the stream cipher algorithm K2 v2.0 [1] that has a dynamic feedback control mechanism.

3.1 Linear Feedback Shift Registers

The K2 v2.0 stream cipher consists of two feedback shift registers (FSRs), *FSR-A* and *FSR-B*, a non-linear function with four internal registers $R1$, $R2$, $L1$, and $L2$, and a dynamic feedback controller as shown in Fig. 1. *FSR-B* is a dynamic feedback shift register. The size of each register is

[1] A previous version of the algorithm is shown in the SASC 2007 workshop record[Kiyomoto et al., 2007].

32 bits. *FSR-A* has five registers, and *FSR-B* has eleven registers. Let β be the roots of the primitive polynomial;

$$x^8 + x^7 + x^6 + x + 1 \in GF(2)[x]$$

A byte string y denotes $(y_7, y_6, ..., y_1, y_0)$, where y_7 is the most significant bit and y_0 is the least significant bit. y is represented by

$$y = y_7\beta^7 + y_6\beta^6 + ... + y_1\beta + y_0$$

In the same way, let γ, δ, ζ be the roots of the primitive polynomials,

$$x^8 + x^5 + x^3 + x^2 + 1 \in GF(2)[x]$$
$$x^8 + x^6 + x^3 + x^2 + 1 \in GF(2)[x]$$
$$x^8 + x^6 + x^5 + x^2 + 1 \in GF(2)[x]$$

respectively.

Let α_0 be the root of the irreducible polynomial of degree four

$$x^4 + \beta^{24}x^3 + \beta^3x^2 + \beta^{12}x + \beta^{71} \in GF(2^8)[x]$$

A 32-bit string Y denotes (Y_3, Y_2, Y_1, Y_0), where Y_i is a byte string and Y_3 is the most significant byte. Y is represented by

$$Y = Y_3\alpha_0^3 + Y_2\alpha_0^2 + Y_1\alpha_0 + Y_0$$

Let α_1, α_2, α_3 be the roots of the irreducible polynomials of degree four

$$x^4 + \gamma^{230}x^3 + \gamma^{156}x^2 + \gamma^{93}x + \gamma^{29} \in GF(2^8)[x]$$
$$x^4 + \delta^{34}x^3 + \delta^{16}x^2 + \delta^{199}x + \delta^{248} \in GF(2^8)[x]$$
$$x^4 + \zeta^{157}x^3 + \zeta^{253}x^2 + \zeta^{56}x + \zeta^{16} \in GF(2^8)[x]$$

respectively.

The feedback polynomials $f_A(x)$, and $f_B(x)$ of *FSR-A* and *FSR-B*, respectively, are as follows;

$$f_A(x) = \alpha_0x^5 + x^2 + 1$$
$$f_B(x) = (\alpha_1^{cl1_t} + \alpha_2^{1-cl1_t} - 1)x^{11} + x^{10} + x^5 + \alpha_3^{cl2_t}x^3 + 1$$

Let $cl1$ and $cl2$ be the sequences describing the output of the dynamic feedback controller. The outputs at time t are defined in terms of some

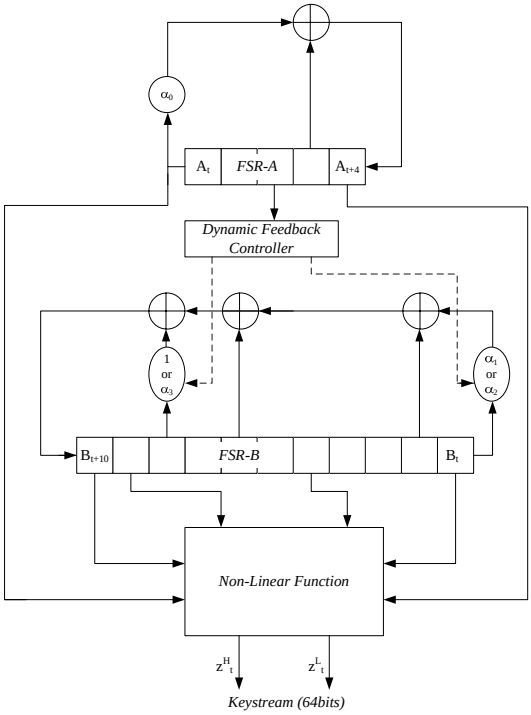

Fig. 1. K2 v2.0 Stream Cipher

bits of FSR-A. Let A_x denote the output of FSR-A at time x, and $A_x[y] = \{0, 1\}$ denote the yth bit of A_x, where $A_x[31]$ is the most significant bit of A_x. Then $cl1$ and $cl2$ (called clock control bits) are described as follows;

$$cl1_t = A_{t+2}[30], \quad cl2_t = A_{t+2}[31]$$

Both $cl1_t$ and $cl2_t$ are binary variables; more precisely, $cl1_t = \{0, 1\}$, and $cl2_t = \{0, 1\}$. Stop-and-go clocking is effective in terms of computational cost, because no computation is required in the case of 0. However, the feedback function has no transformation for feedback registers with a probability 1/4 where all clockings are stop-and-go clockings. Thus, we use two types of clocking for the feedback function. FSR-B is defined by a primitive polynomial, where $cl2_t = 0$.

3.2 Nonlinear Function

The non-linear function of K2 v2.0 is fed the values of two registers of FSR-A and four registers of FSR-B and that of internal registers $R1$, $R2$,

$L1$, $L2$, and outputs 64 bits of the keystream every cycle. Fig. 2 shows the non-linear function of K2 v2.0. The nonlinear function includes four substitution steps that are indicated by Sub.

The Sub step divides the 32-bit input string into four 1-byte strings and applies a non-linear permutation to each byte using an 8-to-8 bit substitution, and then applies a 32-to-32 bit linear permutation. The 8-to-8 bit substitution is the same as s-boxes of AES [Daemen and Rijmen, 1998], and the permutation is the same as AES $Mix\ Column$ operation. The 8-to-8 bit substitution consists of two functions: g and f. The g calculates the multiplicative inverse modulo the irreducible polynomial $m(x) = x^8+x^4+x^3+x+1$ without 0x00, and 0x00 is transformed to itself (0x00). f is an affine transformation defined by;

$$\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \\ 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

where $a = (a_7, ..., a_0)$ is the input and $b = (b_7, ..., b_0)$ is an output, and a_0 and b_0 are the least significant bit (LSB).

Let C be (c_3, c_2, c_1, c_0) and output D be (d_3, d_2, d_1, d_0), where c_i, d_i are 8-bit values. The linear permutation $D = p(C)$ is described as follows;

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

in $GF(2^8)$ of the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.

3.3 Keystream Output

Let keystream at time t be $Z_t = (z_t^H, z_t^L)$ (each z_t^x is a 32-bit value, and z_t^H is a higher string). The keystream z_t^H, z_t^L is calculated as follows:

$$z_t^L = B_t \boxplus R2_t \oplus R1_t \oplus A_{t+4}$$
$$z_t^H = B_{t+10} \boxplus L2_t \oplus L1_t \oplus A_t$$

where A_x and B_x denote outputs of FSR-A and FSR-B at time x, and $R1_x$, $R2_x$, $L1_x$, and $L2_x$ denote the internal registers at time x. The sym-

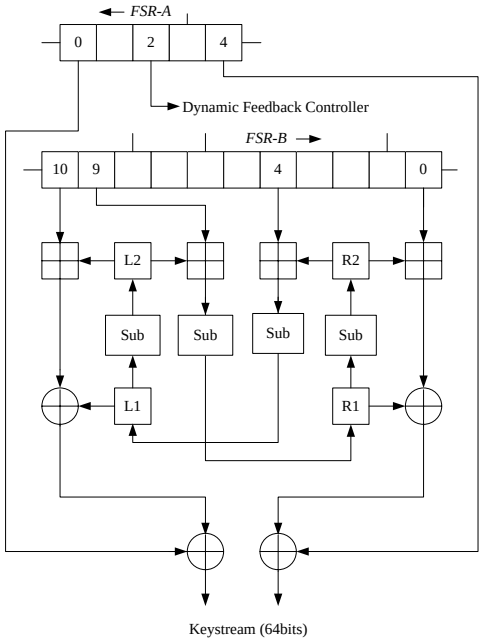

Fig. 2. Non-Linear Function of K2 v2.0

bol $\oplus$ denotes bitwise exclusive-or operation and the symbol $\boxplus$ denotes and 32-bit addition. Finally, the internal registers are updated as follows;

$$R1_{t+1} = Sub(L2_t \boxplus B_{t+9}), \quad R2_{t+1} = Sub(R1_t)$$
$$L1_{t+1} = Sub(R2_t \boxplus B_{t+4}), \quad L2_{t+1} = Sub(L1_t)$$

where $Sub(X)$ is an output of the Sub step for X. The set of $\{B_t, B_{t+3}, B_{t+8}, B_{t+10}\}$ is a *Full Positive Difference Set* (FPDS)[Golic, 1996].

3.4 Initialization Process

The initialization process of K2 v2.0 consists of two steps, a key loading step and an internal state initialization step. First, an initial internal state is generated from a 128-bit initial key, a 192-bit initial key, or a 256-bit initial key and a 128-bit initial vector (IV) by using the key scheduling algorithm. The key scheduling algorithm is similar to the round key generation function of AES and the algorithm extends the 128-bit initial key, the 192-bit initial key or the 256-bit initial key to 384 bits. The key scheduling algorithm for a 128-bit key is described as

$$K_i = \begin{cases} IK_i & (0 \leq i \leq 3) \\ K_{i-4} \oplus Sub((K_{i-1} \lll 8) \oplus (K_{i-1} \ggg 24)) & \\ \oplus Rcon[i/4-1] & (i = 4n) \\ K_{i-4} \oplus K_{i-1} & (i \neq 4n) \end{cases}$$

where $IK = (IK_0, IK_1, IK_2, IK_3)$ is the initial key, i is a positive integer $0 \leq i \leq 11$, and n is a positive integer. The function $Sub(X)$ in the key scheduling algorithm is the same as that in the non-linear function. This function is different from the round key generation function of AES, and the other part of the key scheduling algorithm is same as the AES round key generation. $Rcon[i]$ denotes $(x^i \ mod \ x^8 + x^4 + x^3 + x + 1, 0x00, 0x00, 0x00)$ and x is 0x02. The internal state is initialized with K_i and $IV = (IV_0, IV_1, IV_2, IV_3)$ as follows:

$$\begin{aligned} &A_m = K_{4-m} \ \ (m = 0, ..., 4), B_0 = K_{10}, B_1 = K_{11}, \\ &B_2 = IV_0, B_3 = IV_1, B_4 = K_8, B_5 = K_9, B_6 = IV_2, \\ &B_7 = IV_3, B_8 = K_7, B_9 = K_5, B_{10} = K_6 \end{aligned}$$

The internal registers, $R1$, $R2$, $L1$, and $L2$ are set to 0x00. After the above processes, the cipher clocks 24 times (j = 1, ..., 24), updating the internal states. The internal states A_{j+4} B_{j+10} are also updated as follows:

$$\begin{aligned} A_{j+4} =& \alpha_0 A_{j-1} \oplus A_{j+2} \oplus z_{j-1}^L \\ B_{j+10} =& (\alpha_1^{cl1_{j-1}} + \alpha_2^{1-cl1_{j-1}} - 1) B_{j-1} \oplus B_j \oplus B_{j+5} \\ & \oplus \alpha_3^{cl2_{j-1}} B_{j+7} \oplus z_{j-1}^H \end{aligned}$$

The recommended maximum number of cycles for K2 v2.0 without re-initializing is 2^{58} cycles (2^{64} keystream bits).

4 ANALYSIS OF K2 V2.0

4.1 Analysis of Periods

The cipher has two FSRs. FSR-A is defined by a primitive polynomial. Thus, the sequence of 32-bit outputs generated by FSR-A has a maximum period of $2^{160} - 1$.

Now, we re-define the updating function f_B for FSR-B in terms of a 352×352 matrix M_{f_B} over $GF(2)$. The updating function at time t is given by one of the four possibilities for the matrix. The period of outputs of FSR-B is l, where l is a minimum value satisfying $(M_{f_B})^l = I$. The matrix is described as follows:

$$M_{f_B} = \begin{pmatrix}
0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \\
M_1 & 0 & I & 0 & 0 & I & 0 & 0 & M_2 & 0 & 0
\end{pmatrix}$$

where $M_1 = M_{\alpha_2}, M_{\alpha_1}$ $(cl1_t = 0, 1)$, $M_2 = I, M_{\alpha_3}$ $(cl2_t = 0, 1)$. M_{α_1}, M_{α_2}, and M_{α_3} are shown in Appendix A.

We calculated the periods of FSR-B for the four possible representations as follows:

- $(cl1_t, cl2_t) = (0, 0)$
  The matrix M_{f_B} has the maximum order $2^{352} - 1$, and output sequences of the FSR-B have a maximum period $2^{352} - 1$. The FSR-B is a linear feedback shift register where the feedback polynomial f_B is a primitive polynomial over $GF(2)$.
- $(cl1_t, cl2_t) = (1, 0)$
  The matrix M_{f_B} has the maximum order $2^{352} - 1$, and output sequences of the FSR-B also have the maximum period $2^{352} - 1$.
- $(cl1_t, cl2_t) = (0, 1)$
  The matrix M_{f_B} has an order of approximately 2^{332}, and output sequences of the FSR-B have a period of approximately 2^{332}.
- $(cl1_t, cl2_t) = (1, 1)$
  The matrix M_{f_B} has an order of approximately 2^{318}, and output sequences of the FSR-B have a period of approximately 2^{318}.

From the above results of analysis, we think that K2 v2.0 can produce a keystream of a sufficient length period more than the number of cycles without re-initialization. In an experimental analysis using some sequences of the keystream produced by the cipher, we did not find any short periods.

4.2 Analysis of Statistical Tests

The statistical properties of the cipher also depend on the properties of the output sequences of FSR-A and FSR-B; thus, we expect the keystream of the cipher to have good statistical properties. We evaluated the statistical properties for the keystream of the cipher and output sequences of FSR-A and FSR-B by the NIST Test Suite [2] and confirmed that these properties were good.

4.3 Security Analysis

We discuss security of the cipher against existing attacks.

Time-Memory Trade-Offs. We chose the size of the secret key and IV taking into consideration the discussion of general time-memory trade-offs by Hong and Sarker [Hong and Sarkar, 2005]. This attack generally requires $O(2^{\frac{3(k+v)}{4}})$ pre-computation, $O(2^{\frac{k+v}{2}})$ memory, and $O(2^{\frac{k+v}{2}})$ available data, enabling an online attack with time complexity of $O(2^{\frac{k+v}{2}})$, where the lengths of the secret key and IV are k and v, respectively.

The IV, the secret key, and the internal state are sufficiently large. Thus, we think the cipher is not vulnerable to time-memory trade-off attacks.

Correlation Attacks. The feasibility of correlation attacks depends on the number of inputs to the non-linear function and on the tap positions for the function. The use of a full positive difference set for the tap positions and the non-linear function has sufficient non-linearity for preventing the attacks. We evaluate the security using an asymptotic analysis proposed by Chepyzhov and Smeets [Chepyzhov et al., 2001]. The required length N of the keystream for an correlation attack is $N \approx 1/4 \cdot (2k \cdot h! \cdot ln2)^{1/h} \cdot \epsilon^{-2} \cdot 2^{\frac{l-k}{h}}$, where l, k, and h denote a target FSR length, and algorithm parameters, and the correlation probability of the target stream cipher is $1/2 + \epsilon$. The computational-time complexity of this pre-computation phase in the attack is approximately $N^{\lceil (h-1)/2 \rceil}$ and $N^{\lfloor (h-1)/2 \rfloor}$ is required. Furthermore, the decoding algorithm stores $(N^h \cdot 2^{k-l})/h!$ parity checks and its computational complexity is 2^k times the number of parity checks. When attacking the regular clocked FSR-B in K2, the lowest correlation probability for the attack is approximately

[2] Random Number Generation and Testing, NIST Test Suite, Available from http://csrc.nist.gov/rng/.

$1/2 + 2^{-13}$, where $h = 9$, $k = 26$, and computational complexity and the required memory are roughly $O(2^{256})$. However, no correlation between the input and output sequences of the non-linear function with a probability larger than $1/2 + 2^{-13}$ is found. Furthermore, the irregular clocking improves security against correlation attacks because the linear relations of bits in FSR-B are more complicated using the irregular clock.

Security of the Initialization Process. For any key unique pair of a initial key and a IV, the key loading step generates an internal state of the cipher. The initial key is expanded using the AES key scheduling algorithm, and the IV and expanded keys are thoroughly mixed by the 24 cycles comprising the initialization process. After 13 cycles of the run-up, all values of an internal state of the cipher includes all IV_is. All registers in the internal state will depend on both the initial key and IV after 13 cycles. Furthermore, the initialization process additionally runs 11 cycles and the IV_is and an initial key are well mixed into the internal state. Even an initial difference of any single register for the IV is propagated to all registers in the internal state after 12 steps. Thus, we think that the cipher is not vulnerable to the chosen/related IV attacks. Another attack to the initialization process is a distinguishing attack to use a collision of the first keystream by chosen pairs of an initial key and a IV. However, we did not find the collision that is occurred with a feasible probability, because any registers are well mixed.

Guess-and-Determine Attacks.

First, we discuss the general evaluation of the computational complexity of guess-and-determine attacks. The keystream of n bits is calculated from m bits of the output of an internal state of l bits, which consists of FSRs and the internal memory of a non-linear function. In the initial step, an adversary can determine n bits of the internal state from n bits of the keystream, which allow guessing of $m - n$ bits of the internal state. In the next step, some values of the internal state will have already been guessed or determined in the previous step. In this manner, the adversary guesses other values of the internal state. The adversary iteratively performs guesses and determines steps until all values of the internal state have either been guessed or determined. Now, we assume that m bits of the output are uniformly selected from internal state by an ideal rule. After j steps, the values that the adversary has had to guess can be approximated as $\left(1 - \lceil \frac{v_j}{l} \rceil\right)(m - n)$, where v_j is the number of bits that have already been guessed or determined in step j. Let $y(x)$ denote the

number of already-guessed or determined bits at the xth step of the GD attack. and set $y(0) = 0$. Now, we assume that n bits of the internal state can be determined in each step. $y(x)$ is caluclated as;

$$y(x) = \frac{n^2 - m \cdot n + l \cdot m}{m - n}\left(1 - e^{-\frac{m-n}{l}x}\right)$$

We obtain η the number of steps needed for the GD attack from $y(\eta) = l$. Thus, the total number η of guessed and determined steps can be approximated by $\eta \approx \frac{l}{m-n} \cdot \ln\frac{m}{n}$.

The computational cost C is $C \approx c \cdot 2^{l-n\eta}$, where c is a constant value. For K2, l, m, and n are 640, 256, and 64 respectively. Thus, the computational complexity is approximately $O(2^{344})$. As a result of the general evaluation of GD attacks, K2 is expected to be secure against GD attacks.

A simple guess-and-determine attack is where the attacker guesses all values of FSR-A and all internal memory sets and determines all values of FSR-B. However, this attack is impossible because the computational complexity of the attack is at least $O(2^{288})$. Now, we consider a guess-and-determine attack against a simplified K2 that is performed without multiplying α_i $(i = 0, 1, 2, 3)$ and additions are replaced by exclusive-or operations. First, we consider an attack designed to remove A_{t+4} from the equation of the keystream and to attack with the focus on FSR-B as follows:

$$\begin{aligned} z_t^L \oplus z_{t+4}^H = (B_t \boxplus Sub(R1_{t-1})) \oplus R1_t \\ \oplus (B_{t+14} \boxplus Sub(L1_{t-1})) \oplus L1_{t+4} \end{aligned}$$

If an attacker guesses five elements of the above equation, then the attacker can determine the other element, such as B_{t+14}, and the attacker can also determine A_{t+4}. To determine all values of FSRs, the attacker has to guess at least 10 elements; thus, this attack is impossible. Next, we consider the other attack where the relationship of four internal registers $R1$, $R2$, $L1$, $L2$ is used. The relationship of the memory is described as follows;

$$R2_{t+1} = Sub(R1_t), \;\; L1_{t+2} = Sub(R2_{t+1} \boxplus B_{t+5})$$
$$L2_{t+3} = Sub(L1_{t+2}), \;\; R1_{t+4} = Sub(L2_{t+3} \boxplus B_{t+12})$$

That is, if an attacker guesses $R1_t$, B_{t+5}, B_{t+12}, then the attacker determines $R2_{t+1}$, $L1_{t+2}$, $L2_{t+3}$, $R1_{t+4}$ using the above equations. Now,

consider a more simplified algorithm without FSR-A, which is that the attacker obtains the values of $z_t^H \oplus A_t$ and $z_t^L \oplus A_{t+4}$ in each cycle t. In this situation, if the attacker guesses six elements $R1_{t+1}$, $R1_{t+2}$, $L1_t$, $L1_{t+1}$, B_{t+6}, and B_{t+7}, then the attacker can determine all values of FSR-B. The complexity of the second attack is $O(2^{192})$. However, more than two values of FSR-A have to be guessed for obtaining all values of the internal state. Furthermore, the attacker needs to guess the clock control bits for the full version algorithm. Thus, we think the full version of the algorithm is secure against guess-and-determine attacks.

Distinguishing Attacks. In distinguishing attacks, a probabilistic linear relation of keystream bits is needed as a distinguisher. K. Nyberg and J. Wallen presented a distinguishing attack on SNOW 2.0 [Nyberg and Wallen, 2006] where the computational complexity of their attack was $O(2^{174})$. We try to construct a linear recurrence from output keystream bits with fixed clock control bits $cl1_t = cl2_t = 0$ for each cycle. A two-round linear masking of K2 is shown in Fig. 3. Four substitutions are affected by the linearization; the number of involved substitutions is twice the number of attacks on SNOW 2.0. Thus, we expect that K2 is more secure than SNOW 2.0 against distinguishing attacks.

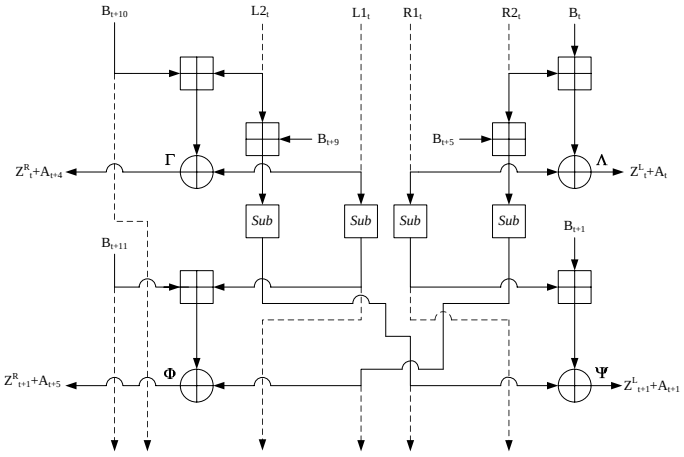

Fig. 3. Linear Masking of K2 for Two-Round Outputs

Now, we construct a linear distinguisher using four masks, Γ, Λ, Φ, and Ψ as follows;

$$
\begin{aligned}
&\Gamma\alpha_2\alpha_0 \cdot \Sigma_{z^H}(t) \oplus \Gamma\alpha_2 \cdot \Sigma_{z^H}(t+3, t+5) \\
&\quad \oplus \Gamma\alpha_0 \cdot \Sigma_{z^H}(t+1, t+6, t+8, t+11) \\
&\quad \oplus \Gamma \cdot \Sigma_{z^H}(t+4, t+6, t+9, t+13, t+14, t+16) \\
&\quad \oplus \Lambda\alpha_2\alpha_0 \cdot \Sigma_{z^L}(t) \oplus \Lambda\alpha_2 \cdot \Sigma_{z^L}(t+3, t+5) \\
&\quad \oplus \Lambda\alpha_0 \cdot \Sigma_{z^L}(t+1, t+6, t+8, t+11) \\
&\quad \oplus \Lambda \cdot \Sigma_{z^L}(t+4, t+6, t+9, t+13, t+14, t+16) \\
&\quad \oplus \Phi\alpha_2\alpha_0 \cdot \Sigma_{z^H}(t+1) \oplus \Phi\alpha_2 \cdot \Sigma_{z^H}(t+4, t+6) \\
&\quad \oplus \Phi\alpha_0 \cdot \Sigma_{z^H}(t+2, t+7, t+9, t+12) \\
&\quad \oplus \Phi \cdot \Sigma_{z^H}(t+5, t+7, t+10, t+14, t+15, t+17) \\
&\quad \oplus \Psi\alpha_2\alpha_0 \cdot \Sigma_{z^L}(t+1) \oplus \Psi\alpha_2 \cdot \Sigma_{z^L}(t+4, t+6) \\
&\quad \oplus \Psi\alpha_0 \cdot \Sigma_{z^L}(t+2, t+7, t+9, t+12) \\
&\quad \oplus \Psi \cdot \Sigma_{z^L}(t+5, t+7, t+10, t+14, t+15, t+17) = 0
\end{aligned}
$$

If the bias for a combination of linear masks has a high probability, an attacker constructs a distinguisher from the equation. However, we have not found a combination of linear masks with a bias value higher than 2^{-128}. Furthermore, to obtain the equation, all clock control bits for 15 times the feedback operations of FSR-B are $cl1_t = cl2_t = 0$; the probability of this condition of clock control bits is about 2^{-30}. That is, the computational complexity of a distinguishing attack against the cipher increase 2^{60} times by using the dynamic feedback control mechanism. Additionally, the cipher is assumed to be re-initialized after 2^{58} cycles. Thus, we conclude that distinguishing attacks against K2 is impossible.

Algebraic Attacks. The non-linear function has ideal algebraic properties; the non-linear function consists of AES S-boxes and an effective permutation function. Furthermore, the dynamic feedback control increases the cost of solving the system of internal values. Courtois presented an evaluation method for the complexity of general algebraic attacks [Courtois, 2005]. A general evaluation suggests that K2 is secure against algebraic attacks; the computational complexity of the attack is roughly $O(2^{646})$.

We investigated the possibility of algebraic attacks, when we assumed that FSR-B has regular clocking and the addition modulo 2^{32} operation is replaced by the XOR operation. An algebraic attack against SNOW 2.0 was proposed by O. Billet and H. Gilbert[Billet and Gilbert, 2005]. We tried to apply the attack to K2. This attack is effective for stream cipher algorithms that have a non-linear function with internal memory.

In this attack, the attacker has to have internal registers at time t, which is defined as linear equations that consist of initial values of internal state variables, and constructs relationships between the input values of a non-linear substitution and the corresponding output values, which are low degree algebraic expressions.

First, we obtain the following equation from the assumption:

$$R2_t = R1_t \oplus A_{t+4} \oplus B_t \oplus z_t^L, \;\; L1_{t-1} = Sub(R2_{t-2} \oplus B_{t+2})$$

$$R1_t = Sub(L1_{t-1} \oplus A_{t-1} \oplus B_{t+8} \oplus B_{t+9} \oplus z_{t-1}^H)$$

If we remove substitutions from the above equations, we obtain the linear recurrence $R2_t = R2_{t-2} \oplus A_{t-1} \oplus A_{t+4} \oplus B_t \oplus B_{t+2} \oplus B_{t+8} \oplus B_{t+9} \oplus z_{t-1}^H \oplus z_t^L$.

We can define a linear equation of registers of FSR-A and FSR-B, for any clock t, which only involves keystream, registers at $t = 0$, and $R2_0$ from the equation for $R2_t$;

$$R2_t = R2_0 \bigoplus_{i=0}^{t} \epsilon_t^i z_i^H \bigoplus_{j=0}^{t} \epsilon_t^j z_j^L \bigoplus_{k=0}^{4} \epsilon_t^k A_k \bigoplus_{l=0}^{10} \epsilon_t^l B_l$$

where ϵ_t^i, ϵ_t^j, ϵ_t^k, ϵ_t^l are known binary coefficients. We also obtain;

$$R1_{t-1} = R1_0 \bigoplus_{i=0}^{t} \epsilon_t^i z_i^H \bigoplus_{j=0}^{t} \epsilon_t^j z_j^L \bigoplus_{k=0}^{4} \epsilon_t^k A_k \bigoplus_{l=0}^{10} \epsilon_t^l B_l$$

in the same manner. Thus, we can obtain quadratic equations to use the relation $R2_t = Sub(R1_{t-1})$ because the substitution consists of the AES S-Box, which has linearly independent quadratic equations involving the S-Box input and output bits.

However, the substitutions that were removed in the above attack prevent construction of the linear equations for $R1_t$ and $R2_t$ in the real algorithm. Thus, the attacker cannot obtain the linear equation of the fixed values of internal memory and registers, $R1_0$, $R2_0$, B_0, B_1, ..., B_{10}, A_0, ..., A_4 for $R1_{t-1}$ and $R2_t$. This attack is impossible even for the regular clocking algorithm.

Furthermore, the attacker has to guess the clocks of each cycle to determine the equations for a full version of the cipher. Let M be the total number of non-constant monomials appearing in the over-defined system of equations, and N be the number of equations that the attacker obtains per output of one cycle. The computational complexity of the algebraic attack increases $2^2 \cdot (\lceil M/N \rceil - 1)$ $(\leq 2^{160})$ times by using the

dynamic feedback control mechanism. Thus, we think the full version algorithm is secure against an algebraic attack.

Clock Control Guessing Attack. This attack [Zenner, 2003] is effective against bit-oriented clock controlled stream ciphers. K2 is a word-oriented stream cipher with a large internal state, and its non-linear part is more complicated than existing stream ciphers broken by the attacks. An extended attack based on an algebraic approach was discussed by S. Al-Hinai *et. al.* [Al-Hinai et al., 2006]. However, it is difficult to apply the attack when a sufficiently secure non-linear function is used to generate the keystream. Thus, we expect that the cipher will be secure against such attacks.

Divide-and-Conquer Approach. The output sequences of FSR-A and FSR-B have good statistical properties. Thus, we expect that divide-and-conquer attacks for the FSRs are not feasible.

4.4 Performance Analysis

We implemented the algorithm on a PC (Pentium 4 3.2 GHz) using Intel C++ Compiler Ver.9 (for Windows), and evaluated the performance. The results of the evaluation are shown in Table 1. "Key. Gen." indicates the required clock-cycles for a one-byte keystream generation and "Init." indicates the required clock-cycles for one initialization, including the initial key and IV setup. The optimal version is optimized to produce a 128-byte keystream at once. The performance of eSTREAM [3] candidates is about 4–14 cycle/byte in software implementation. The performance of K2 v2.0 is much faster than existing clock controlled stream ciphers and AES, and is competitive against word-oriented stream ciphers. K2 v2.0 is improves the security against existing attacks proposed for SNOW 2.0. The *Inner State Efficiency* (ISE) [Zenner, 2004] of the cipher, 0.4, is sufficiently efficient.

5 CONCLUSION

This paper proposed a new design for a stream cipher, which is a word-oriented stream cipher using dynamic feedback control. The stream cipher K2 v2.0 is secure against several different types of attacks, and it offers

[3] eSTREAM Project, http://www.ecrypt.eu.org/stream/

Table 1. Comparison with Other Stream Ciphers

Algorithm	Structure	Key. Gen. (Cy./By.)	Init. (Cy./Init.)	ISE
SNOW 2.0[Ekdahl and Johansson, 2003]	LFSR	4.5	937	0.440
SOBER-t32[NESSIE, 2003]	LFSR	28	944	0.290
LILI-128[NESSIE, 2003]	CC-LFSR	987	59	0.375
RC4 128-bit key [NESSIE, 2003]	Table Update.	20	4680	0.018
AES 128-bit key encryption [NESSIE, 2003]	-	24	689	-
K2 v2.0 (Reference)	DFSR	7.5	1308	0.400
K2 v2.0 (Optimal)	DFSR	5.4	1136	0.400

high-performance encryption and decryption for software implementations. Furthermore, the design of K2 v2.0 is considered security against existing attacks on SNOW 2.0. We believe that the dynamic feedback control mechanism is potentially effective against several different types of attacks, not only existing attacks but also novel attacks.

References

[Al-Hinai et al., 2006] Al-Hinai, S., Batten, L., Colbert, B., and Wong, K. (2006). Algebraic attacks on clock-controlled stream ciphers. In *Proc. of ACISP 2006, LNCS*, volume 4058, pages 1–16. Springer Verlag.

[Babbage and Dodd, 2005] Babbage, S. and Dodd, M. (2005). The stream cipher MICKEY-128 2.0. *Article for eSTREAM Project, available at* http://www.ecrypt.eu.org/stream/p2ciphers/mickey128/mickey128_p2.pdf.

[Billet and Gilbert, 2005] Billet, O. and Gilbert, H. (2005). Resistance of SNOW 2.0 against algebraic attacks. In *Proc. of CT-RSA 2005, LNCS*, volume 3376, pages 19–28. Springer Verlag.

[Chepyzhov et al., 2001] Chepyzhov, V., Johansson, T., and Smeets, B. (2001). A simple algorithm for fast correlation attacks on stream ciphers. In *Proc. of FSE'00, LNCS*, volume 1978, pages 181–195. Springer Verlag.

[Courtois, 2005] Courtois, N. (2005). Algebraic attacks on combiners with memory and several outputs. In *Proc. of ICISC 2004, LNCS*, volume 3506, pages 3–20. Springer Verlag.

[Daemen and Rijmen, 1998] Daemen, J. and Rijmen, V. (1998). *The Design of Rijndael, Information Security and Cryptography, Texts and Monographs.* Springer Verlag.

[Ekdahl and Johansson, 2000] Ekdahl, P. and Johansson, T. (2000). Snow -a new stream cipher. *The NESSIE submission paper.*

[Ekdahl and Johansson, 2003] Ekdahl, P. and Johansson, T. (2003). A new version of the stream cipher SNOW. In *Proc. of SAC 2002, LNCS*, volume 2595, pages 47–61. Springer Verlag.

[Golic, 1996] Golic, J. D. (1996). On security of nonlinear filter generators. In *Proc. of FSE '96, LNCS*, volume 1039, pages 173–188. Springer Verlag.

[Hong and Sarkar, 2005] Hong, J. and Sarkar, P. (2005). Rediscovery of time memory tradeoffs. *IACR ePrint Archive, Report 2005/090.*

[Jansen et al., 2005] Jansen, C. J., Helleseth, T., and Kholosha, A. (2005). Cascade jump controlled sequence generator and POMARANCH stream cipher. *Article for eSTREAM Project, available at* http://www.ecrypt.eu.org/stream/p2ciphers/pomaranch/pomaranch_p2.pdf.

[Kiyomoto et al., 2007] Kiyomoto, S., Tanaka, T., and Sakurai, K. (2007). A word-oriented stream cipher using clock control. In *SASC 2007 Workshop Record*, pages 260–274, available at `http://sasc.crypto.rub.de/files/sasc2007_record.zip`.

[NESSIE, 2003] NESSIE (2003). Nessie final report, parformance of optimized implementations of the nessie primitives. *NES/DOC/TEC/WP6/D21/2.*

[Nyberg and Wallen, 2006] Nyberg, K. and Wallen, J. (2006). Improved linear distinguishers for SNOW 2.0. In *Proc. of FSE 2006, LNCS*, volume 4047, pages 144–162. Springer Verlag.

[Rose and Hawkes, 1999] Rose, G. and Hawkes, P. (1999). The t-class of sober stream cipher. *Publication Document, QUALCOMM Australia.*

[Simpson et al., 2000] Simpson, L., Dawson, E., Golic, J., and Millan, W. (2000). LILI keystream generator. In *Proc. of SAC 2000, LNCS*, volume 2012, pages 248–261. Springer Verlag.

[Zenner, 2003] Zenner, E. (2003). On the efficiency of the clock control guessing attack. In *Proc. of ICISC'02, LNCS*, volume 2587, pages 200–212. Springer Verlag.

[Zenner, 2004] Zenner, E. (2004). On the role of the inner state size in stream ciphers. *Reihe Informatik 01-2004.*

A Matrix of α_i

M_{α_1} is the 32×32 matrix over $GF(2)$ given by

$$\begin{pmatrix}
10001101100000000000000000000000\\
11000110010000000000000000000000\\
11100011001000000000000000000000\\
11111100000100000000000000000000\\
11111110000010000000000000000000\\
01110010000001000000000000000000\\
00110100000000100000000000000000\\
00011010000000010000000000000000\\
00110111000000001000000000000000\\
10011011000000000100000000000000\\
01001101000000000010000000000000\\
10010001000000000001000000000000\\
11001000000000000000100000000000\\
11010011000000000000010000000000\\
11011110000000000000001000000000\\
01101111000000000000000100000000\\
00100111000000000000000010000000\\
10010011000000000000000001000000\\
11001001000000000000000000100000\\
01000011000000000000000000010000\\
10100001000000000000000000001000\\
01110111000000000000000000000100\\
10011100000000000000000000000010\\
01001110000000000000000000000001\\
01010100000000000000000000000000\\
10101010000000000000000000000000\\
01010101000000000000000000000000\\
01111111000000000000000000000000\\
10111111000000000000000000000000\\
00001011000000000000000000000000\\
01010001000000000000000000000000\\
10101000000000000000000000000000
\end{pmatrix}$$

M_{α_2} is the 32×32 matrix over $GF(2)$ given by

$$
\begin{pmatrix}
01010010100000000000000000000000 \\
10101001010000000000000000000000 \\
00000110001000000000000000000000 \\
10000011000100000000000000000000 \\
11000001000010000000000000000000 \\
00110010000001000000000000000000 \\
01001011000000100000000000000000 \\
10100101000000010000000000000000 \\
01110011000000001000000000000000 \\
00111001000000000100000000000000 \\
01101111000000000010000000000000 \\
00110111000000000001000000000000 \\
00011011000000000000100000000000 \\
01111110000000000000010000000000 \\
11001100000000000000001000000000 \\
11100110000000000000000100000000 \\
00100110000000000000000010000000 \\
10010011000000000000000001000000 \\
11101111000000000000000000100000 \\
01110111000000000000000000010000 \\
10111011000000000000000000001000 \\
01111011000000000000000000000100 \\
10011011000000000000000000000010 \\
01001101000000000000000000000001 \\
01011101000000000000000000000000 \\
00101110000000000000000000000000 \\
01001010000000000000000000000000 \\
00100101000000000000000000000000 \\
00010010000000000000000000000000 \\
01010100000000000000000000000000 \\
01110111000000000000000000000000 \\
10111011000000000000000000000000
\end{pmatrix}
$$

M_{α_3} is the 32×32 matrix over $GF(2)$ given by

$$\begin{pmatrix}
10011010100000000000000000000000\\
01001101010000000000000000000000\\
00111100001000000000000000000000\\
10000100000100000000000000000000\\
11000010000010000000000000000000\\
11100001000001000000000000000000\\
01101010000000100000000000000000\\
00110101000000010000000000000000\\
00000010000000001000000000000000\\
00000001000000000100000000000000\\
10000010000000000010000000000000\\
01000011000000000001000000000000\\
00100001000000000000100000000000\\
00010000000000000000010000000000\\
00001010000000000000001000000000\\
00000101000000000000000100000000\\
10110010000000000000000010000000\\
11011001000000000000000001000000\\
11011110000000000000000000100000\\
11011101000000000000000000010000\\
11101110000000000000000000001000\\
11110111000000000000000000000100\\
11001001000000000000000000000010\\
01100100000000000000000000000001\\
11001101000000000000000000000000\\
01100100000000000000000000000000\\
11001101000000000000000000000000\\
11100110000000000000000000000000\\
00111110000000000000000000000000\\
11010010000000000000000000000000\\
11101001000000000000000000000000\\
11110100000000000000000000000000\\
00110111000000000000000000000000\\
10011011000000000000000000000000
\end{pmatrix}$$

B Test Vector

A test vector for K2 is shown in this section.

Initial Key (128 bits):
IK_0 =0x00000000, IK_1 =0x00000000,
IK_2 =0x00000000, IK_3 =0x00000000

Initial Vector (128 bits):
IV_0 =0x00000000, IV_1 =0x00000000,
IV_2 =0x00000000, IV_3 =0x00000000

Keystream:
0xF871EBEF945B7272, 0xE40C04941DFF0537,
0x0B981A59FBC8AC57, 0x566D3B02C179DBB4,
0x3B46F1F033554C72, 0x5DE68BCC9872858F,
0x575496024062F0E9, 0xF932C998226DB6BA